Lab. nr 1

Wprowadzenie. Podstawy programowania Arduino

Program zajęć:

1. Wprowadzenie do Arduino

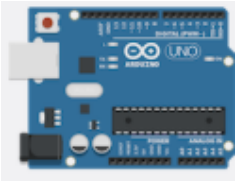
Na zajęciach będzie wykorzystywana płytka Arduino Uno R3 lub jej klony (w wersji fizycznej lub z symulatora).

Obecnie można korzystać z dwóch symulatorów:

- https://www.tinkercad.com/dashboard
- https://wokwi.com

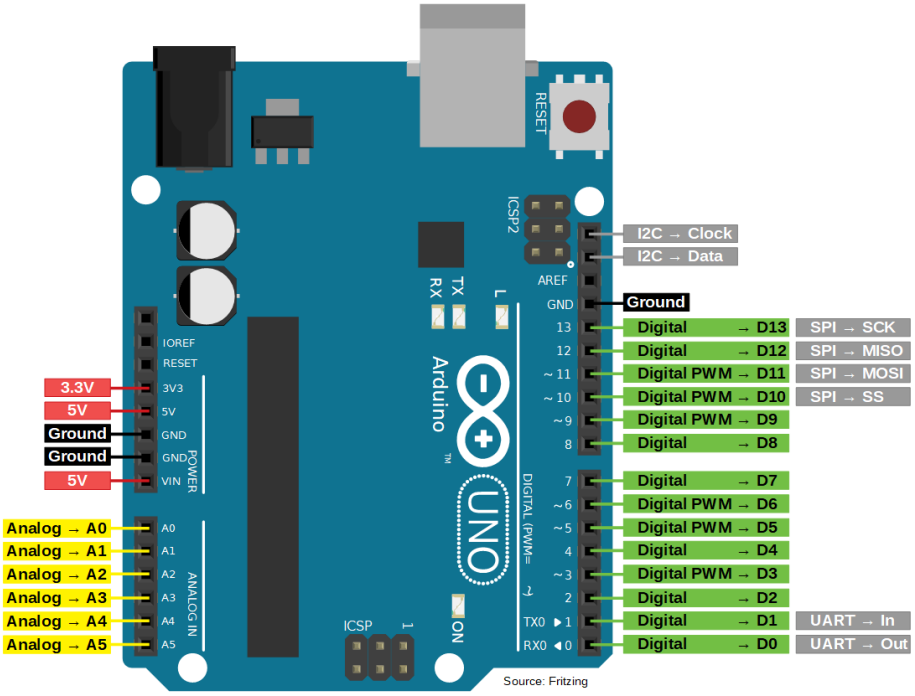
Ogólna charakterystyka komponentów, które wykorzystamy do zadań laboratoryjnych:

Arduino Uno R3 – programowalna płytka firmy Arduino w wersji trzeciej. Posiada na pokładzie mikrokontroler ATmega328P. Posiada 14 cyfrowych pinów I/O (wejścia/wyjścia) (z których 6 można wykorzystać jako wyjścia PWM) oraz 6 wejść analogowych. Arduino wykorzystuje komunikacje poprzez interfejs USB. Płytkę Arduino programujemy poprzez Aduino IDE za pomocą języka C/C++. W naszym przypadku wykorzystamy symulator Arduino (dzięki temu możemy testować proste programy bez potrzeby posiadania fizycznej płytki).

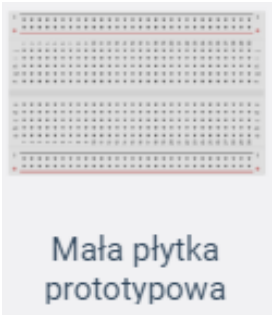


Arduino Uno R3

Poniżej rozmieszczenie pinów w Arduino Uno R3.

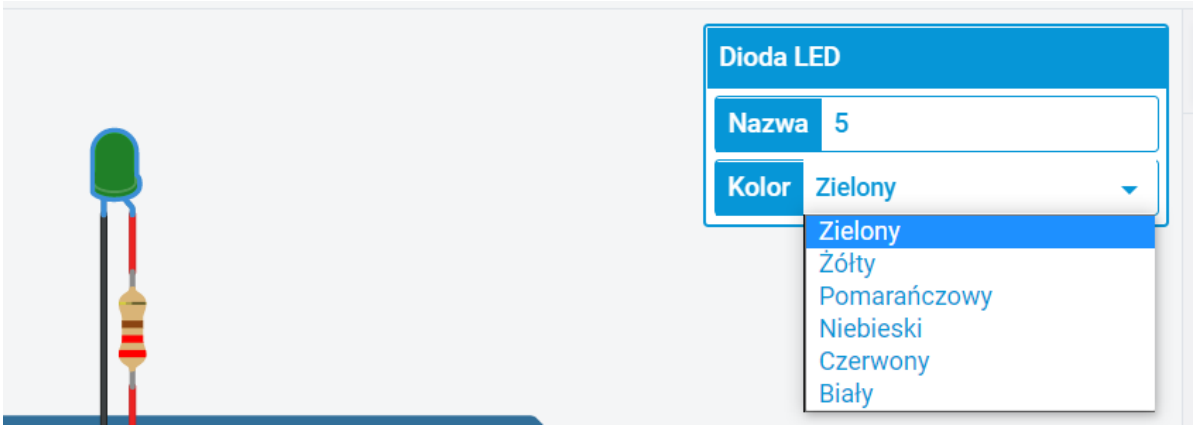


Płytka stykowa – jest używana przede wszystkim do budowania i testowania prototypów zaprojektowanych obwodów. Dzięki niej można łatwo testować i nanosić ewentualne poprawki, bez konieczności lutowania ze sobą komponentów.



Dioda LED – Standardowa dioda. Symulator Tinkercad umożliwia użycie diod w 6 kolorach.

Aby zmienić kolor diody, w momencie, kiedy umieścimy ją na schemacie, trzeba „kliknąć w nią" i pojawia nam się dodatkowe menu wyboru (poniższy zrzut).



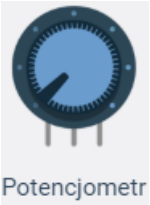
Rezystor – program umożliwia zdefiniowanie wartości rezystora, jaką tylko potrzebujemy. Odpowiednio po zadanej wartości zmieniają się kolory pasków.

W celu zmiany wartości rezystancji należy kliknąć w rezystor, kiedy znajduję się już w obwodzie i zdefiniować potrzebną nam wartość (poniższy zrzut).

Czujnik Temperatury – analogowy czujnik temperatury. Umożliwia pomiar temperatury w zakresie od -40°C do +125°C.

Przycisk monostabilny (Tact Switch) – przycisk, który w chwili kliknięcia (dopóki nie puścimy go) posiada stan wysoki (HIGH), po puszczeniu przycisk wraca do stanu niskiego (LOW).

Potencjometr – element wyposażony w pokrętło. Potencjometr działa na zasadzie dzielnika napięcia. Najpopularniejszym zastosowaniem potencjometrów jest regulacja prądu lub napięcia w urządzeniach elektrycznych. Wykorzystanie zostanie do zmiany częstotliwości świecenia ekranu LCD.

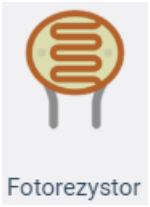


Serwomechanizm – układ sterowania, którego sygnałem wejściowym może być położenie, prędkość oraz przyspieszenie.

Dioda RGB – Dioda posiadająca trzy wyprowadzenia (czerwoną, zieloną i niebieską) oraz wspólną katodę. Taka dioda może świecić w jednym głównym kolorze, jak również można mieszać intensywności świecenia w celu uzyskania wybranego przez nas dowolnego koloru. Dioda RGB nadaje się do stosowania równych efektów wizualnych.


Dioda LED RGB

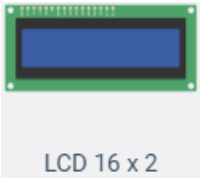
Fotorezystor – element, którego opór jest zależny od natężenia światła. Posiada dwie nóżki oraz powierzchnię światłoczułą.


Fotorezystor

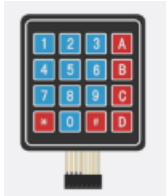
Czujnik odległościowy – czujnik odbierający sygnały ultradźwiękowe. Umożliwia precyzyjne wykrywanie przeszkód.


Ultradźwiękowy czujnik odległości

Ekran LCD – Wyświetlacz, który składa się z dwóch wierszy. W każdym wierszu znajduję się 16 pikseli do wyświetlania informacji. Wyświetlacz posiada 16 pinów.

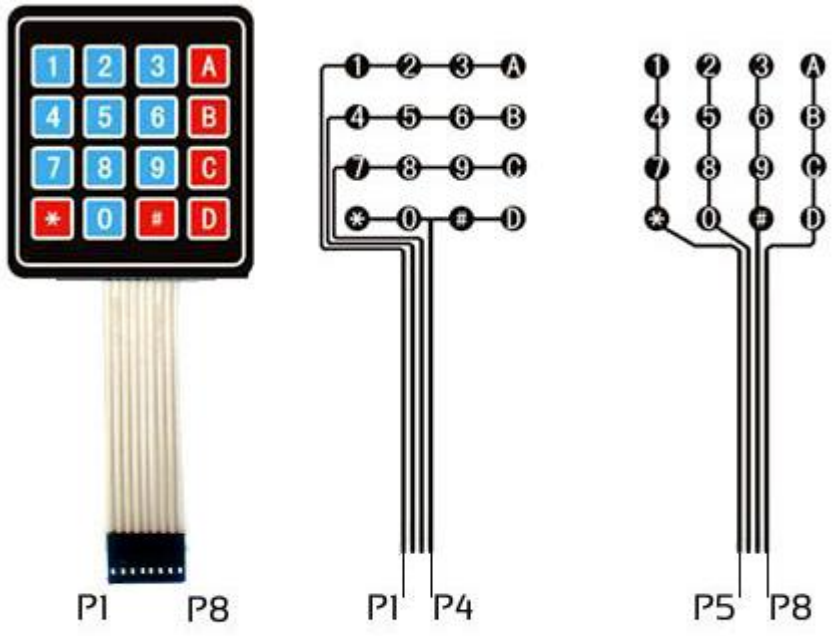

LCD 16 x 2

Klawiatura - klawiatura numeryczna wyposażona w cyfry: od 0 do 9, litery: A, B, C, D oraz znaki: * i #.

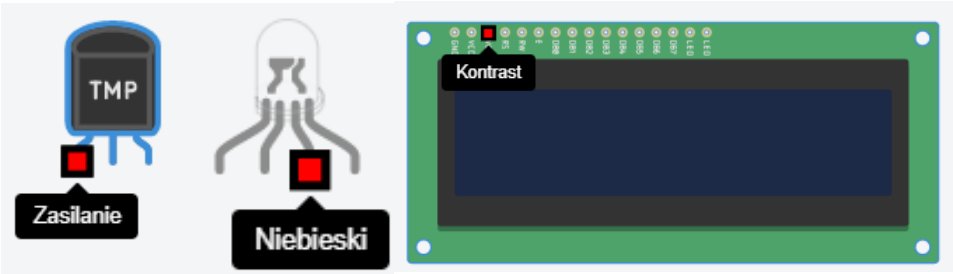


Klawiatura 4x4

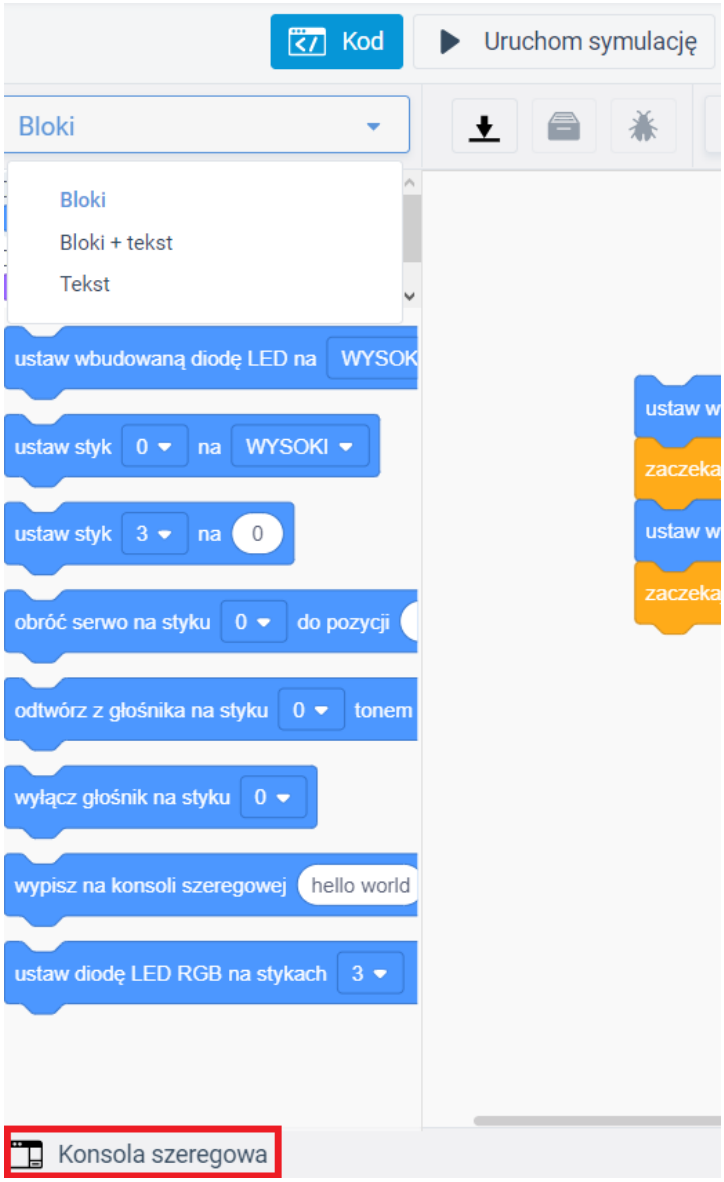
Klawiaturę membranową można podłączyć na dwa sposoby.



Wskazując kursorem na odpowiednią „nóżkę” komponentu możemy sprawdzić, do którego pinu została ona podłączona.



W celu wprowadzeniu kodu klikamy po prawej stronie „Kod”. Pierwotnie pojawia nam się możliwość wprowadzania kodu za pomocą bloków. Po rozwinięciu listy można zmienić sposób wprowadzania kodu za pomocą tekstu (do wszystkich zadań będziemy korzystać z tekstowego sposobu implementacji kodu).



W tym samym okienku na dole również mamy możliwość włączenia podglądu wyświetlanych danych w monitorze portu szeregowego.

Następnie uruchamiamy naszą symulację i obserwujemy działanie/poprawiamy błędy.

2. Programowanie Arduino

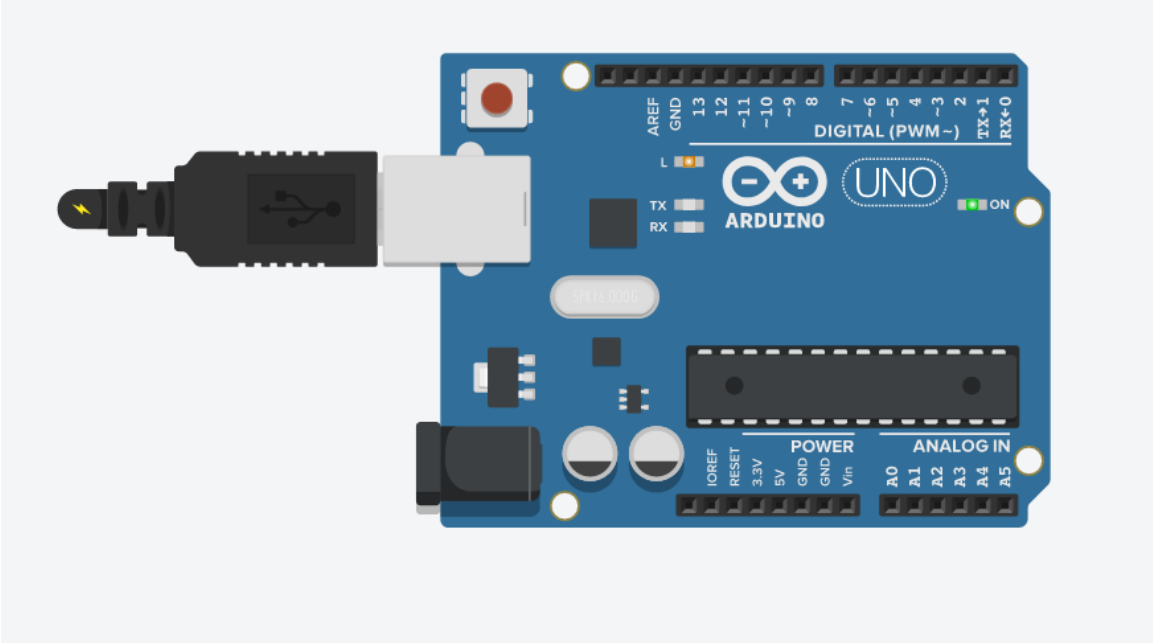
a) Pierwszy program

Zaczniemy od przetestowania czy nasze Arduino jest sprawne. Mając fizyczne urządzenie, możemy pobrać dedykowane IDE, a następnie po podłączeniu go do komputera, możemy w IDE Arduino wybrać przykładowy program (Plik -> Przykłady -> 01 Basic -> Blink).

Aczkolwiek zadania laboratoryjne zostały przygotowane do wykonania w sposób symultaniczny z wykorzystaniem symulatora internetowego.

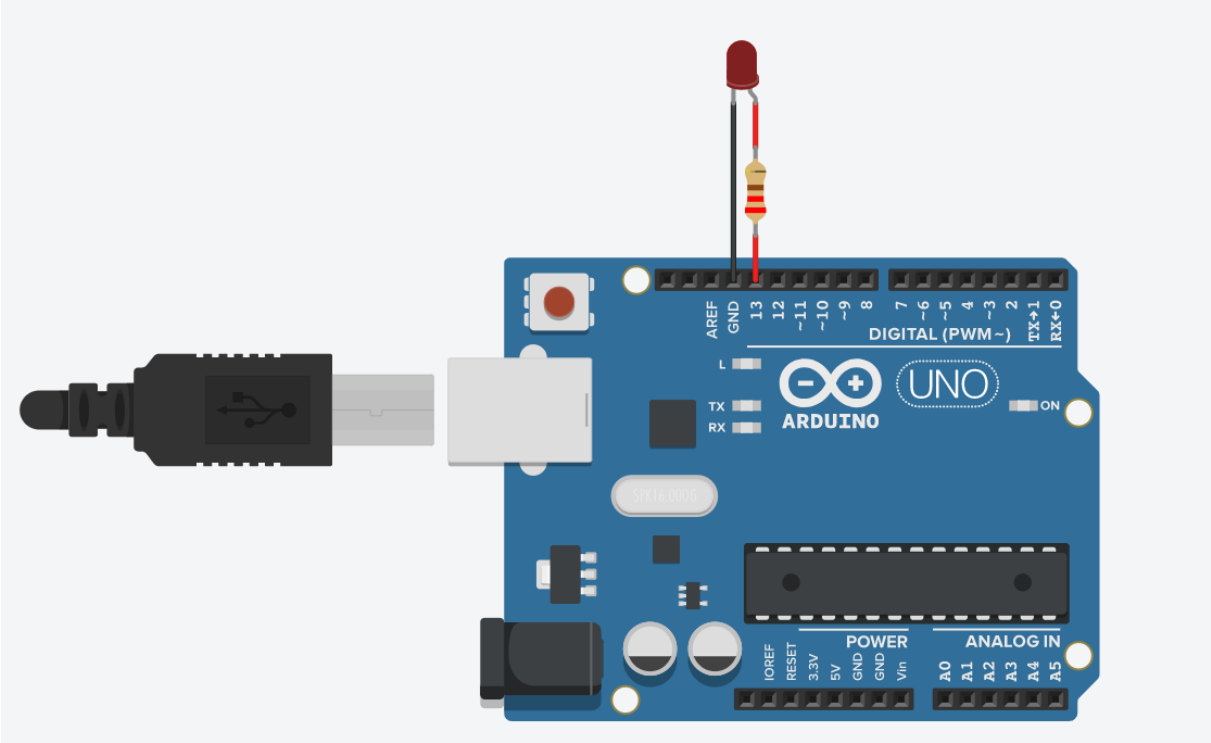
W symulatorze wystarczy wkleić poniższy kod:

```
void setup()
{
      pinMode(LED_BUILTIN, OUTPUT);
}
void loop()
{
      digitalWrite(LED_BUILTIN, HIGH);
      delay(1000);
      digitalWrite(LED_BUILTIN, LOW);
      delay(1000);
}
```

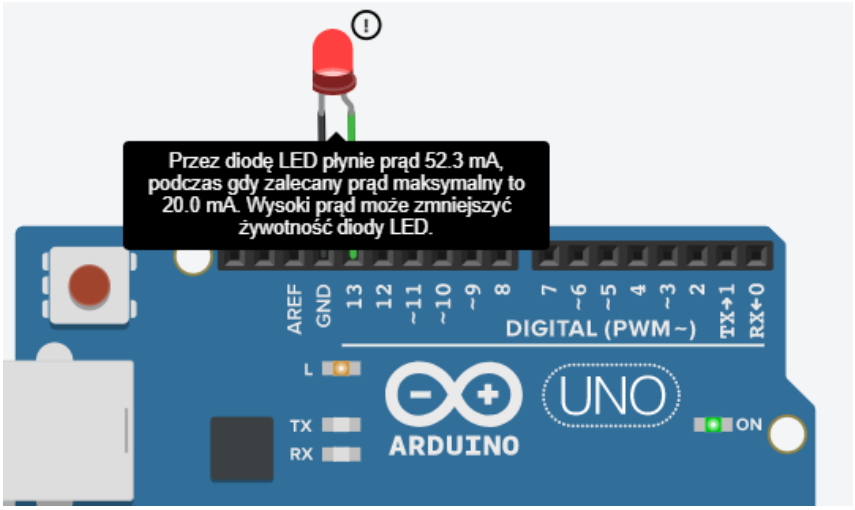


Na powyższym rysunku można zauważyć zapaloną diodę na płytce (pod pinem GND). Dioda włącza się i wyłącza co 1 sek. Dzięki temu wiemy, że nasze Arduino działa poprawnie.

b) Miganie diodą

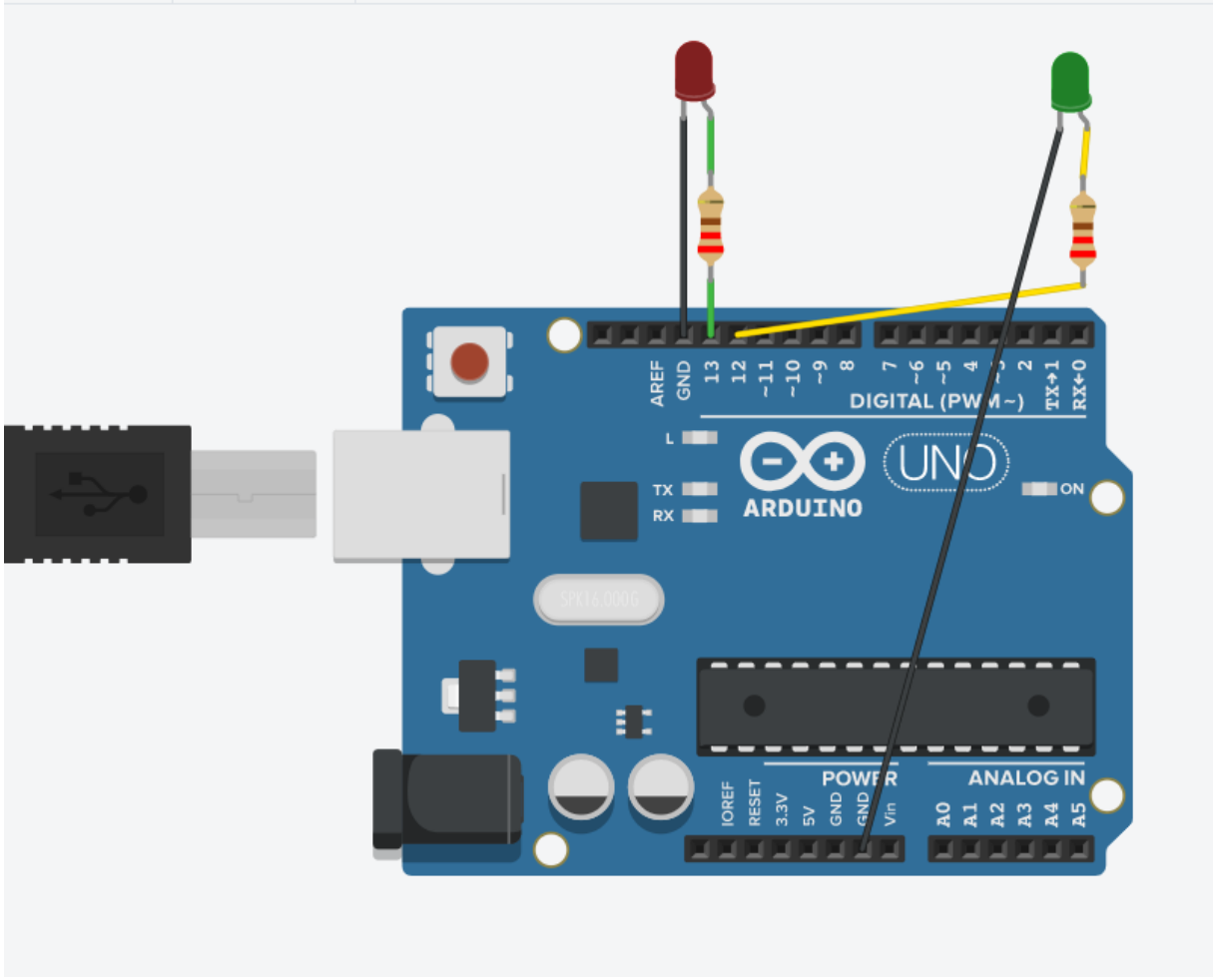


W programie wykorzystano jedną diodę oraz rezystor 220Ω. Dioda co sekundę włącza się oraz wyłącza. Może się zdarzyć sytuacja, że zapomnimy o użyciu rezystora (tu albo w przyszłych programach). Tutaj przewagę ma symulator nad fizycznym sprzętem. Robiąc na fizycznym sprzęcie zapewne taka dioda by się spaliła. W symulatorze natomiast dostajemy ostrzeżenie. Dzięki temu wiemy, że zapomnieliśmy wykorzystać rezystor.



```
void setup()
{
  pinMode(13, OUTPUT);
}
void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

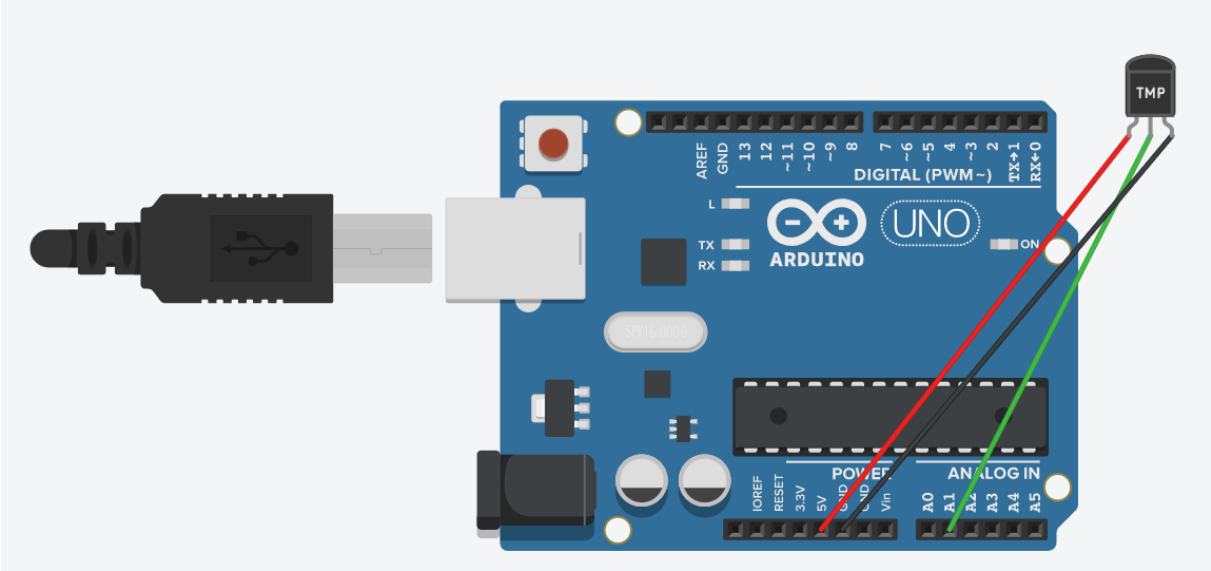
c) Miganie dwoma diodami



Ten program jest rozbudowaną wersją poprzedniego programu. Zastosowano w nim dwie diody różnego koloru. Diody będą się zapalać i wyłączać na zmianę.

```
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  digitalWrite(12, HIGH);
}
void loop()
{
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  delay(1000);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  delay(1000);
}
```

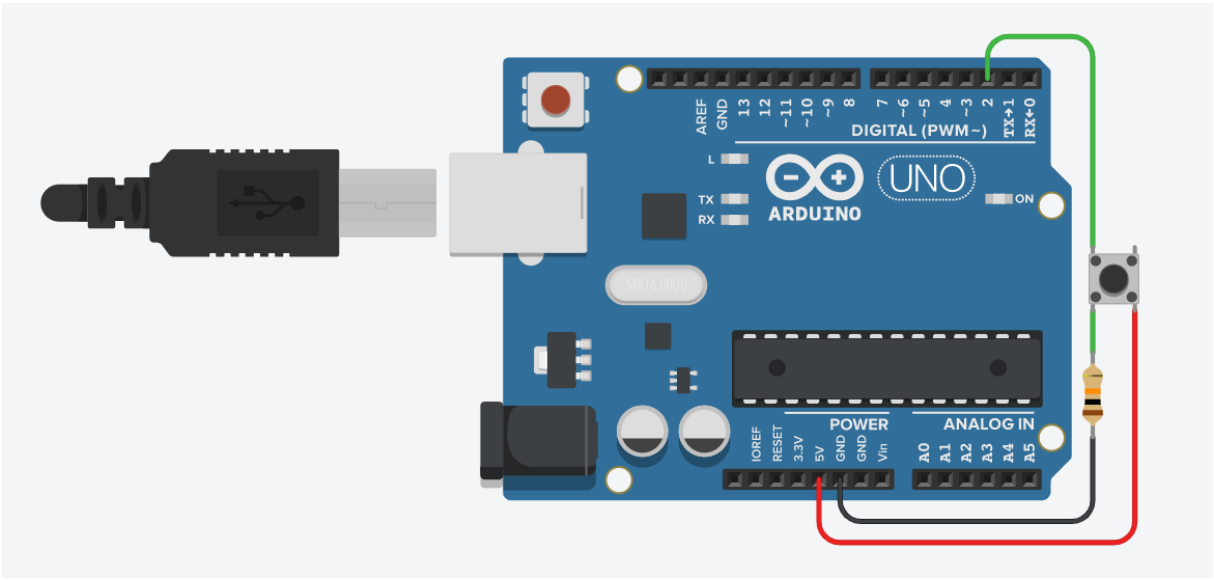
d) Pomiar temperatury



W tym programie został wykorzystany analogowy czujnik temperatury (cyfrowy niestety nie jest dostępny w symulatorze). Program realizuje odczyt wartości, którą następnie przeliczamy na wartość napięcia. Otrzymaną wartość konwertujemy na temperaturę, a następnie wypisujemy wartość w monitorze portu szeregowego.

```
int czujnik = A1;   //pin analogowy A1 połączony z sygnałem z
czujnika
float VOLT;
float TEMP;
void setup(){
  Serial.begin(9600);        //inicjalizacja monitora szerego-
wego
  Serial.println("Test czujnika temperatury");
}
void loop(){
  int odczyt = analogRead(czujnik);      //odczytanie wartości
z czujnika
  VOLT = (odczyt * 5.0) / 1024.0; //przeliczenie odczytanej
wartości na napięcie                                    w wol-
tach (dla podłączenia pod 5 V)
  TEMP = (VOLT - 0.5) * 100;  //konwersja z napięcia na tempe-
raturę, rozdzielczość         czujnika wynosi 10 mV na stopień,
dodatkowo należy zastosować offset 500 mV
  Serial.print("Temperatura (C): ");     //wyświetlenie jej na
monitorze
  Serial.println(TEMP);
  delay(200);                    //opóźnienie między kolejnymi
odczytami
}
```

e) Przycisk chwilowy



Program ten przedstawia sposób działania przycisku monostabilnego.

```
int buttonState = 0;

void setup()
{
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  buttonState = digitalRead(2);
  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
  delay(10);
}
```

3. Zadania do samodzielnej realizacji

a) Zad.1. Sygnalizacja czasowa
Zbuduj symulator świateł drogowych. Światło czerwone i zielone mają się palić przez 30 sek. Światło pomarańczowe przełączające się pomiędzy czerwonym i zielonym ma trwać 3 sek.

a) Zad. 2. Sygnalizacja warunkowa
Przerób program z zadania 1, aby zmieniał sygnalizację za pomocą przycisku chwilowego. Każde kliknięcie przycisku ma zmieniać kolor diody – wejściowo niech się pali jedna z głównych diod sygnalizacji (czerwona bądź zielona). W momencie kliknięcia przycisku, dioda włączona ma się wyłączyć, a włączyć się mają dwie pozostałe diody, jednak dioda pomarańczowa ma po chwili zgasnąć.

a) Zad. 3. Cyfrowy termometr
Zaprojektuj obwód i zaimplementuj program z użyciem termometru, diody RGB i pojedynczej diody (lub z wykorzystaniem 4 osobnych diod różnego koloru). Dioda ma zmieniać kolor zależny od wprowadzonych zakresów temperaturowych. Dla temperatury poniżej 0 ma się świecić niebieski kolor, dla wartości od 0 do 10^{o}C kolor żółty, od 10^{o}C do 20^{o}C kolor zielony i od 20^{o}C wzwyż kolor czerwony.

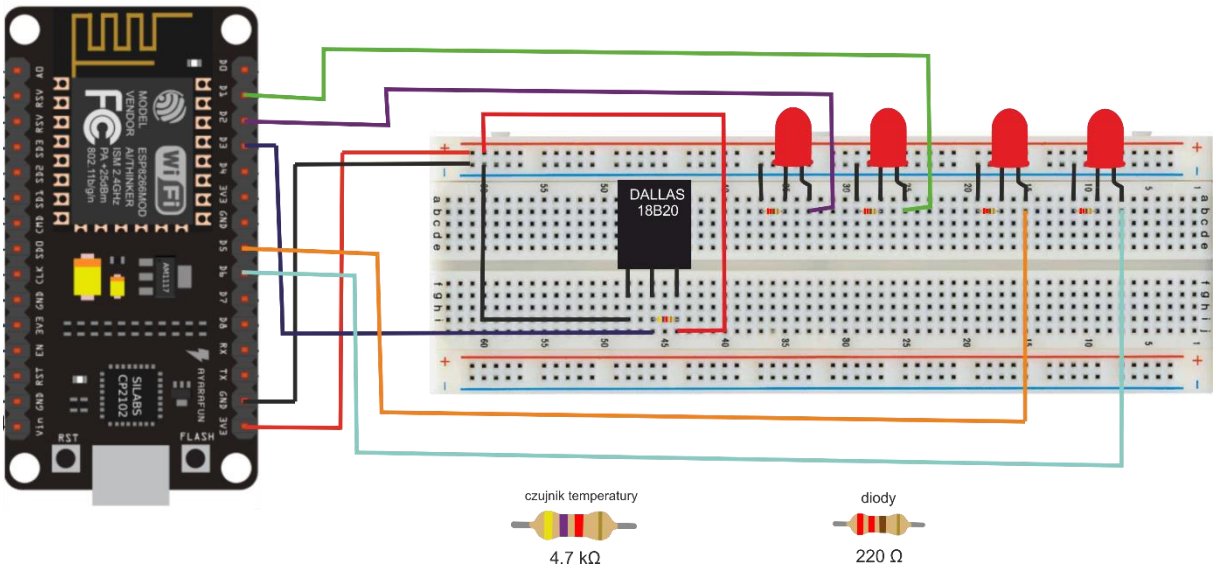
W sprawozdaniu proszę umieścić schematy połączeń oraz kod programu

4. Przydatne źródła

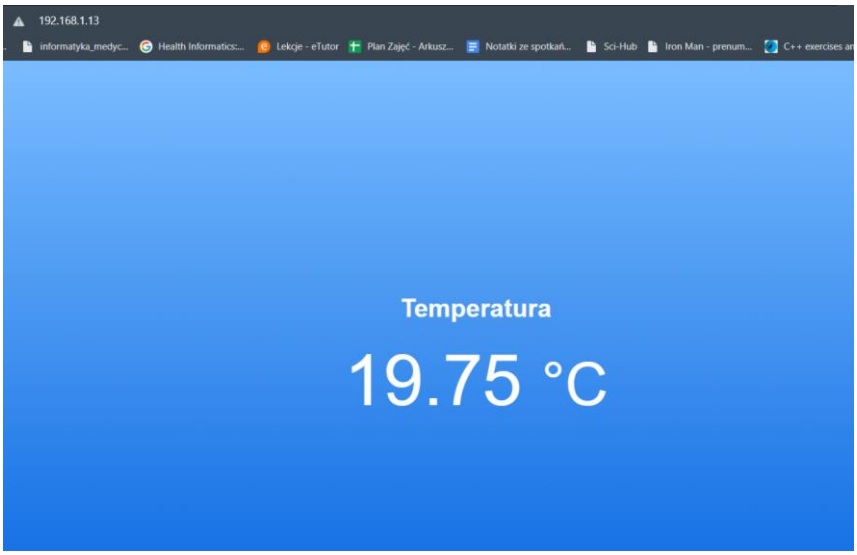
Dokumentacja ATmega328P - http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
Podstawowy kurs Arduino - https://forbot.pl/blog/kurs-arduino-podstawy-programowania-spis-tresci-kursu-id5290
Rozszerzony kurs Arduino - https://forbot.pl/blog/kurs-arduino-ii-wstep-spis-tresci-id15494
Oficjalna strona Arduino - https://www.arduino.cc
Blog o Arduino - http://www.jarzebski.pl/arduino.html
Simon Monk. Arduino dla początkujących. Podstawy i szkice.
Simon Monk. Arduino dla początkujących. Kolejny krok.
Kimmo Karvinen, Tero Karvinen. Czujniki dla początkujących. Poznaj otaczający Cię świat za pomocą elektroniki, Arduino i Raspberry Pi.

5. Dla chętnych

Przykład rozwiązania zadania trzeciego z wykorzystaniem cyfrowego termometru Dallas ds18b20 oraz informacją na prostym serwerze.



Rysunek przedstawia prosty schemat połączenia. W projekcie wykorzystano płytkę NodeMCU, ze względu na posiadanie na swoim pokładzie modułu Wi-Fi.



Kod programu:

```
#include <OneWire.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DallasTemperature.h>


#define ONE_WIRE_BUS D3


OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature DS18B20(&oneWire);


//Dane WIFI

const char* ssid = "PLAY INTERNET 4G LTE-F928";
const char* password = "93642682";


ESP8266WebServer server(80);


//Termometr
```

```
char temperatureString[6];
const int led = 13;


float getTemperature() {
  float temp;
  do {
    DS18B20.requestTemperatures();
    temp = DS18B20.getTempCByIndex(0);
    delay(100);
     }
while (temp == 85.0 || temp == (-127.0));

return temp;
}



//DIODA

String output5State = "off";
String output4State = "off";
String output14State = "off";
String output12State = "off";

const int output5 = 5; //Pin D1
const int output4 = 4; //Pin D2
const int output14 = 14; //Pin D5
const int output12 = 12; //Pin D6


// obecny czas

unsigned long currentTime = millis();

// poprzedni czas

unsigned long previousTime = 0;

// definicja czasu w milisekundach

const long timeoutTime = 2000;

////////////////////////////////////

void setup(void){

  Serial.begin(115200);



  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);
```

```
pinMode(output14, OUTPUT);
pinMode(output12, OUTPUT);

digitalWrite(output5, LOW);
digitalWrite(output4, LOW);
digitalWrite(output14, LOW);
digitalWrite(output12, LOW);



WiFi.begin(ssid, password);
Serial.println("");



while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}


Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());



server.on("/", []() {
  float temperature = getTemperature();
  dtostrf(temperature, 2, 2, temperatureString);


  String title = "Temperatura";
  String cssClass = "mediumhot";


    if (temperature < 25)
   {
    cssClass = "cold";
    output4State = "off";
    output5State = "off";
    output14State = "off";
    digitalWrite(output4, LOW);
    digitalWrite(output5, LOW);
    digitalWrite(output14, LOW);
    }

    else if(temperature > 25 && temperature < 27)
   {
    cssClass = "cold";
```

```
  output4State = "on";
  output5State = "off";
  output14State = "off";
  digitalWrite(output4, HIGH);
  digitalWrite(output5, LOW);
  digitalWrite(output14, LOW);
  }

  else if (temperature > 27 && temperature < 29)
  {
  cssClass = "mediumhot";
  output4State = "on";
  output5State = "on";
  output14State = "off";
  digitalWrite(output5, HIGH);
  digitalWrite(output4, HIGH);
  digitalWrite(output14, LOW);
  }

 else if (temperature > 29 && temperature < 31)
  {
  cssClass = "mediumhott";
  output4State = "on";
  output5State = "on";
  output14State = "on";
  digitalWrite(output4, HIGH);
  digitalWrite(output5, HIGH);
  digitalWrite(output14, HIGH);
  }

else if (temperature > 31)
  {
  cssClass = "hot";
  output4State = "off";
  output5State = "off";
  output14State = "off";
  digitalWrite(output4, LOW);
  digitalWrite(output5, LOW);
  digitalWrite(output14, LOW);



  digitalWrite(output12, HIGH);
  delay(5);
  digitalWrite(output12, LOW);
  delay(5);
  }

//HTML
```

```
    String message = "<!DOCTYPE html><html><head><title>" + title +
"</title><meta charset=\"utf-8\" /><meta name=\"viewport\" con-
tent=\"width=device-width\" /><style>\n";

    message += "html {height: 100%;}";

    message += "div {color: #fff;font-family: 'Arial';font-weight:
400;left: 50%;position: absolute;text-align: center;top:
50%;transform: translateX(-50%) translateY(-50%);}";

    message += "h2 {font-size: 90px;font-weight: 400; margin: 0}";

    message += "body {height: 100%;}";

    message += ".cold {background: linear-gradient(to bottom,
#7abcff, #0665e0 );}";

    message += ".mediumhot {background: linear-gradient(to bottom,
#81ef85,#057003);}";

    message += ".mediumhott {background: linear-gradient(to bottom,
#fe7f00, #ff8413);}";

    message += ".hot {background: linear-gradient(to bottom,
#fcdb88,#d32106);}";

    message += "</style>";

    message += "<meta http-equiv=\"refresh\" content=\"1\">";

    message += "</head><body class=\"" + cssClass + "\"><div><h1>" +
title +  "</h1><h2>" + temperatureString +
" <small>&deg;C</small></h2></div></body></html>";


    server.send(200, "text/html", message);

  });

  server.begin();

  Serial.println("Start strony!");

}

void loop(void){

  server.handleClient();

}
```